A Short Introduction of Behavior Tree

SHANG-CHING LIU

ADD SLIDE NUMBERS

Outline

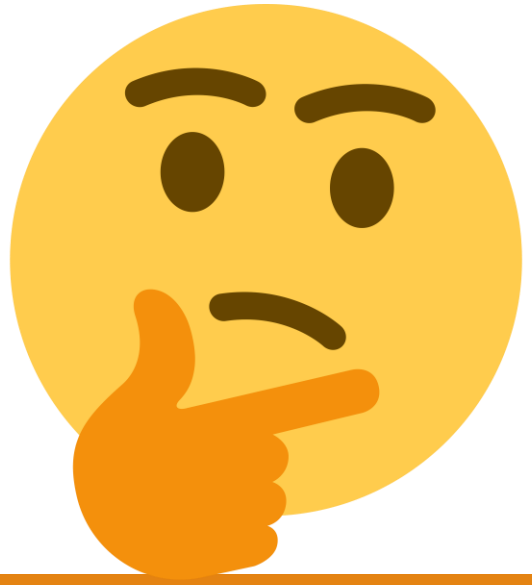
1. Motivation
2. Tools and example
3. Create behavior tree
4. Behavior Trees and Machine Learning overview
5. Q & A
6. Reference

Motivation

Emm mm…. What should the robot do next?

Giving initial state(S0) and target state(S*) and the restriction space of plan(π) in between, planning the best action path.

$$\pi(S^*, S0) = \text{Action}$$



The image is capture from the book[1].

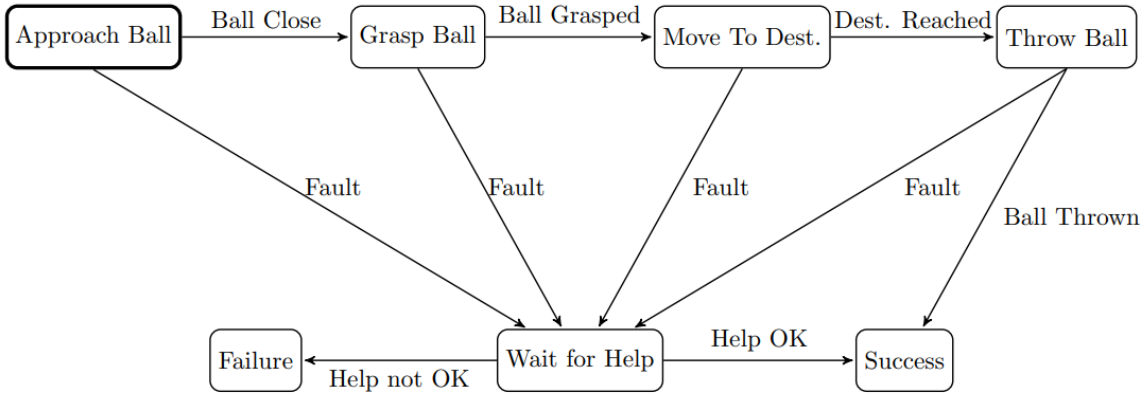
Tools and example

How can we formalize our thought?

1. Script
   - Straight Forward

2. Finite State Machine (FST)
   - Structural way to describe the thought
   - High dependency between related states
   - None-People Character(NPC)

Script

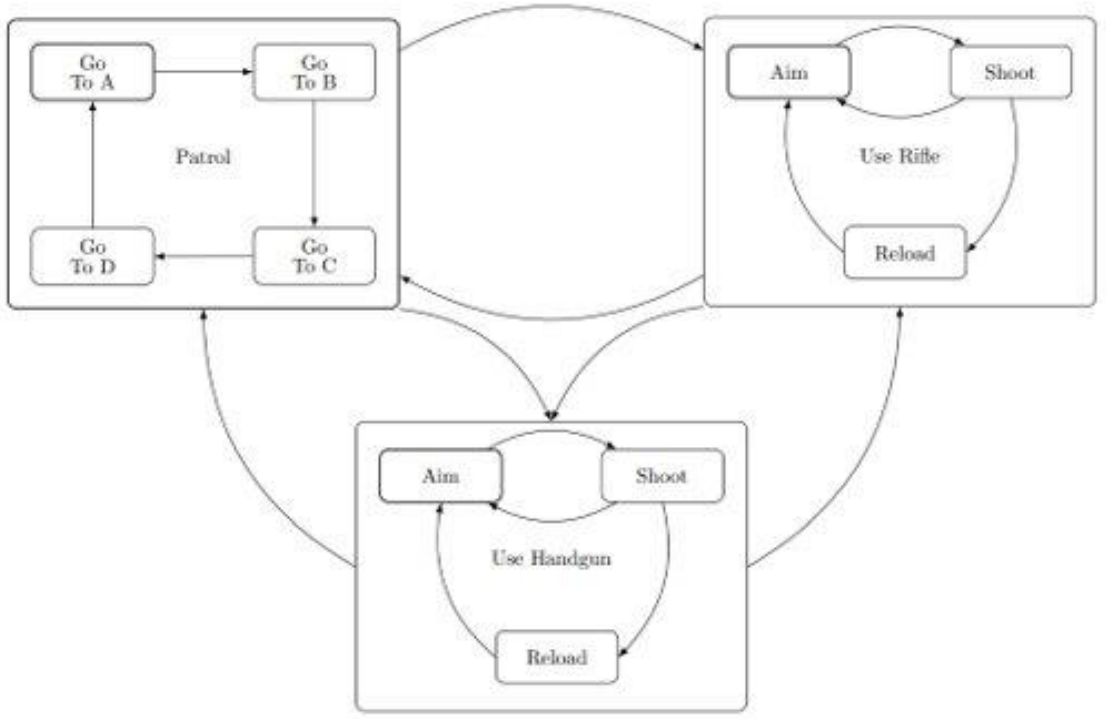


Finite State Machine

[The image is capture from the book[1].]

How can we formalize our thought?

3. Hierarchical Finite State Machine (HFST)
    - Decouple by making the layer and subgroup.
    - Inside each subgroup is another FST.

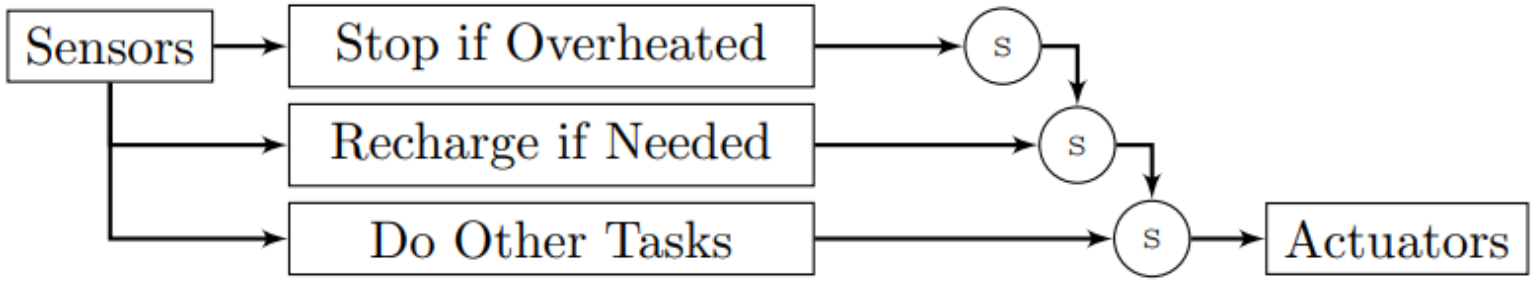


Hierarchical Finite State Machine

[The image is capture from the book[1].]

How can we formalize our thought?

4. Subsumption Architecture
    - Parallel
    - Prioritize
    - Hard to scale and maintain

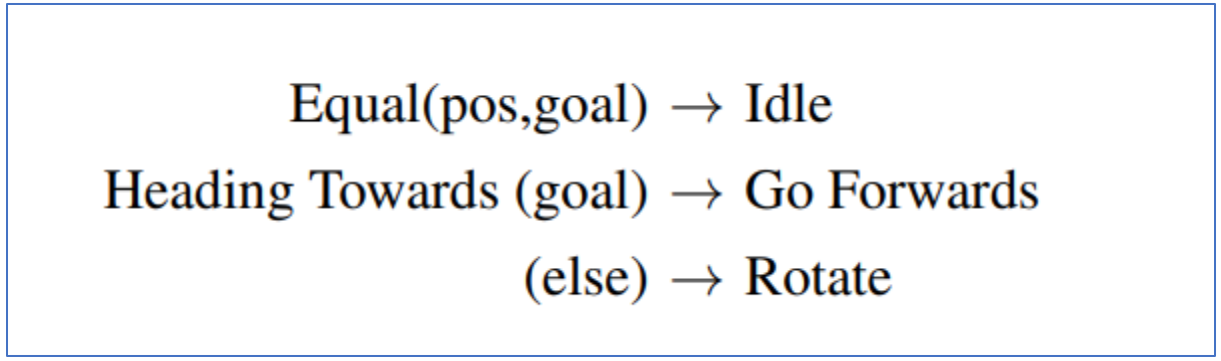


Subsumption Architecture

[The image is capture from the book[1].]

How can we formalize our thought?

5. Teleo-Reactive(TR)
   - Reactive
   - Easy to design
   - Hard to maintain and handle failure

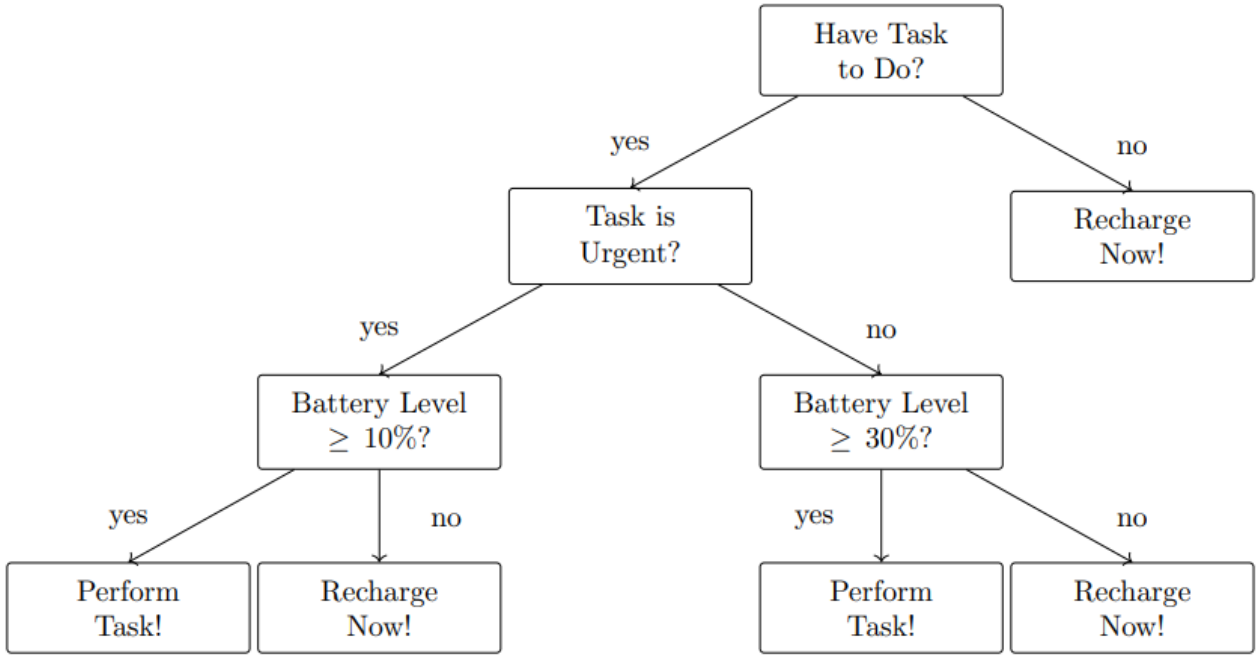


Subsumption Architecture

[The image is capture from the book[1].]

How can we formalize our thought?

6. Decision Tree
   - Modularity
   - Hierarchy
   - Easy to design
   - Hard to handle failure due to no information flow in between

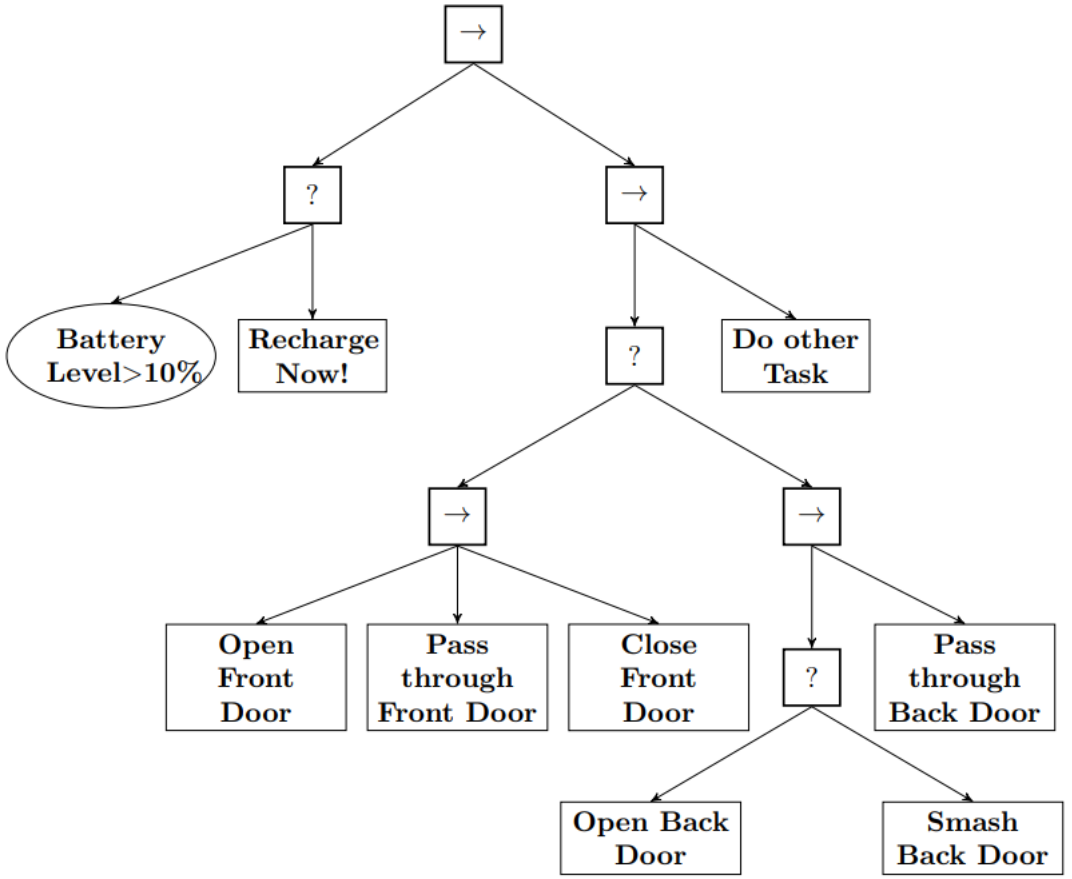


Decision Tree

[The image is capture from the book[1].]

How can we formalize our thought?

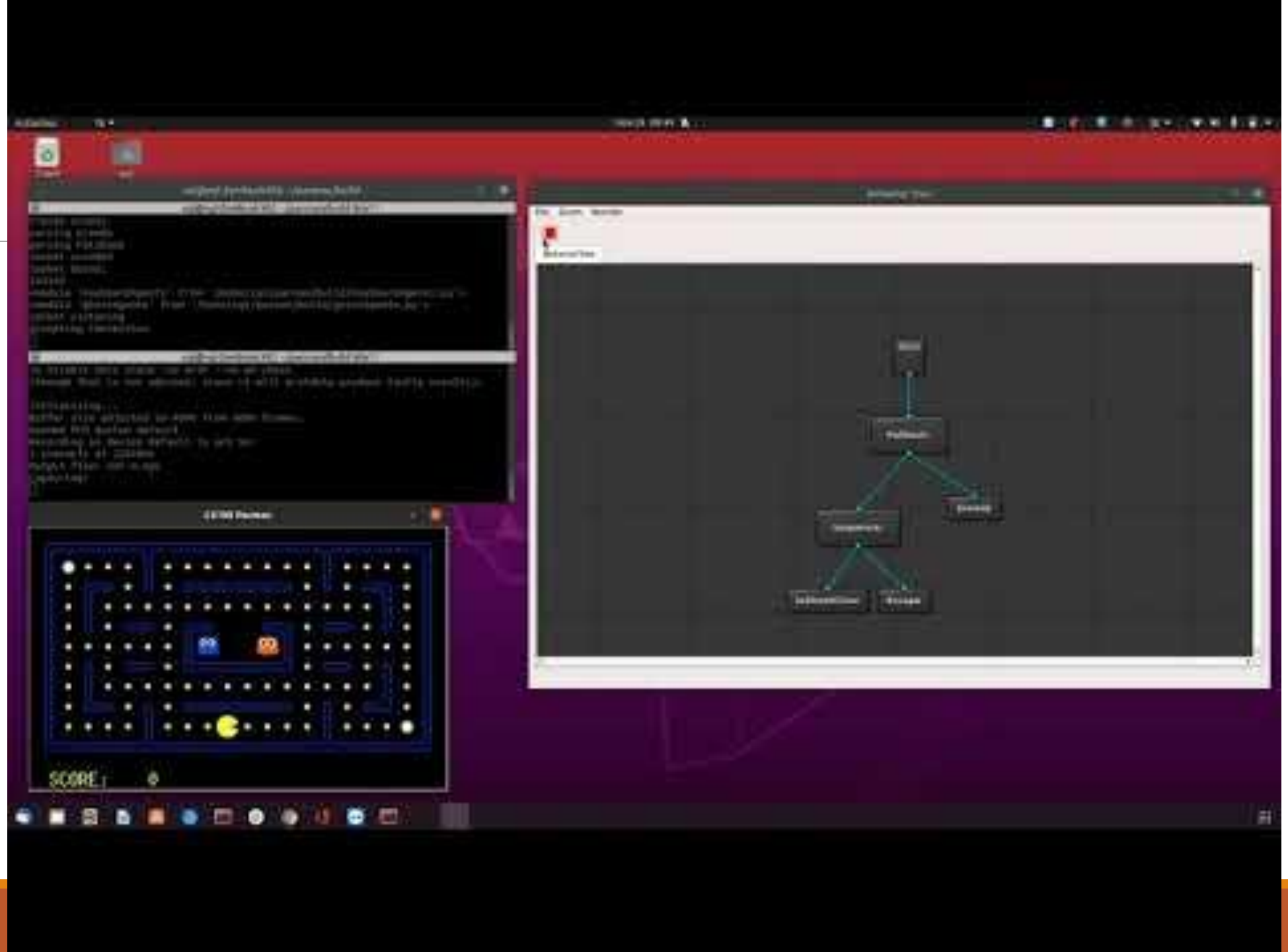
7. Behavior Tree (BT)
   - Structural
   - Can easy collapse each function as a node
   - Reactiveness and Modularity
   - Hard to think in BT



Behavior Tree

[The image is capture from the book[1].]

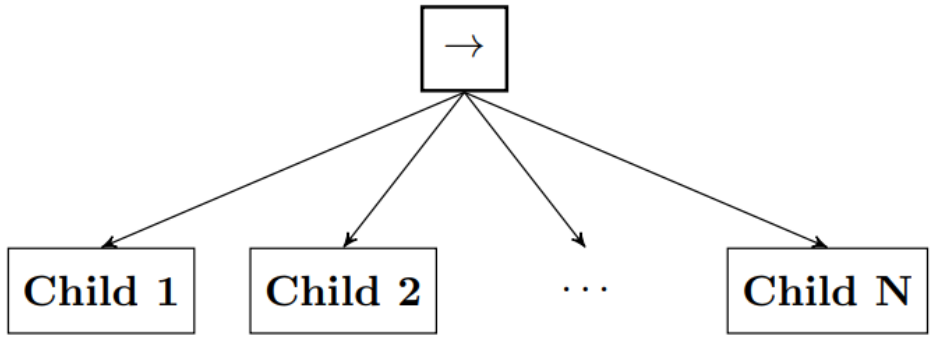
Pacman Example [2]



Create Behavior Tree

Elements in Behavior Tree

- Sequence Node
  - Return: Running, Failure, Success
  - From left to right
  - Return success when all the sub-node return success



Sequence Node

[The image is capture from the book[1].]

Algorithm 1: Pseudocode of a Sequence node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick (child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Failure then
6         return Failure
7 return Success
```

Sequence Node Algorithm

[The image is capture from the book[1].]

Elements in Behavior Tree

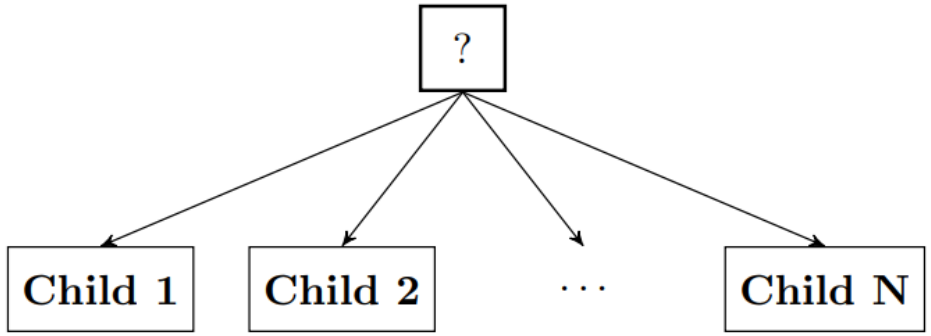
- Fallback Node
  - Return: Running, Failure, Success
  - Return Success and Running if any of the node return Success and Running
  - Only return failure when all sub-nodes return failure

Algorithm 2: Pseudocode of a Fallback node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick (child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Success then
6         return Success
7 return Failure
```

Fallback Node Algorithm

[The image is capture from the book[1].]



Fallback Node

[The image is capture from the book[1].]

Elements in Behavior Tree

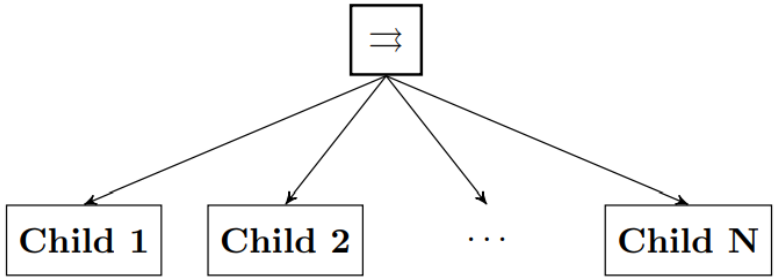
- Parallel Node
  - Return: Running, Failure, Success
  - Return Success if more than M of children return success
  - Return Failure if less than M of children return success
  - Otherwise return Running

Algorithm 3: Pseudocode of a Parallel node with N children and success threshold M

```
1 for i ← 1 to N do
2     childStatus(i) ← Tick(child(i))
3 if Σ_{i:childStatus(i)=Success} 1 ≥ M then
4     return Success
5 else if Σ_{i:childStatus(i)=Failure} 1 > N − M then
6     return Failure
7 return Running
```

Parallel Node Algorithm

[The image is capture from the book[1].]

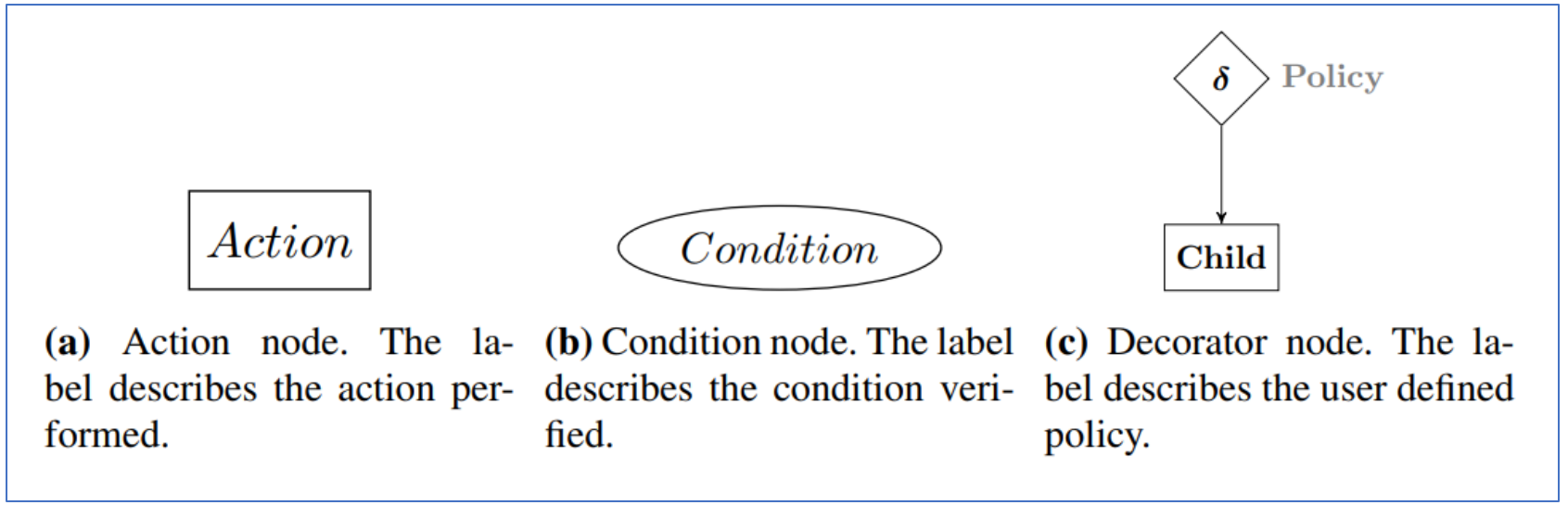


Parallel Node

[The image is capture from the book[1].]

Elements in Behavior Tree

- Action Node
- Condition Node
- Decorator Node



Other Nodes

[The image is capture from the book[1].]

Elements in Behavior Tree

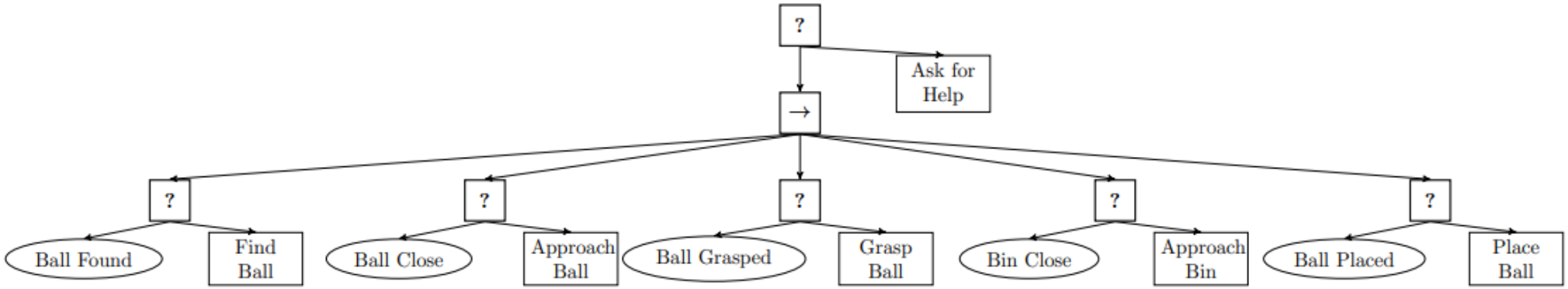
Node type	Symbol	Succeeds	Fails	Running
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Sequence	$\rightarrow$	If all children succeed	If one child fails	If one child returns Running
Parallel	$\rightrightarrows$	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	text	Upon completion	If impossible to complete	During completion
Condition	text	If true	If false	Never
Decorator	$\Diamond$	Custom	Custom	Custom

Summary

[The image is capture from the book[1].]

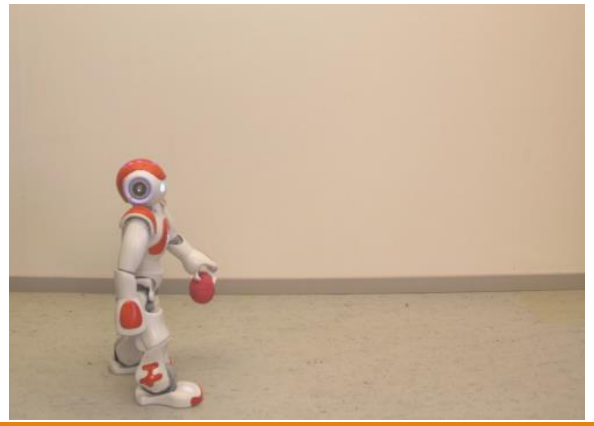
Elements in Behavior Tree

- Traversal example



Example

[The image is capture from the book[1].]



Ball Grasped

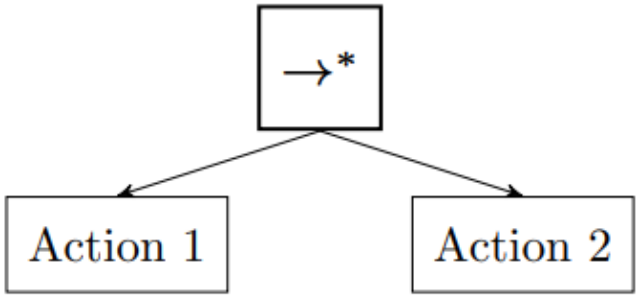
[The image is capture from the book[1].]

Ball being taken

[The image is capture from the book[1].]

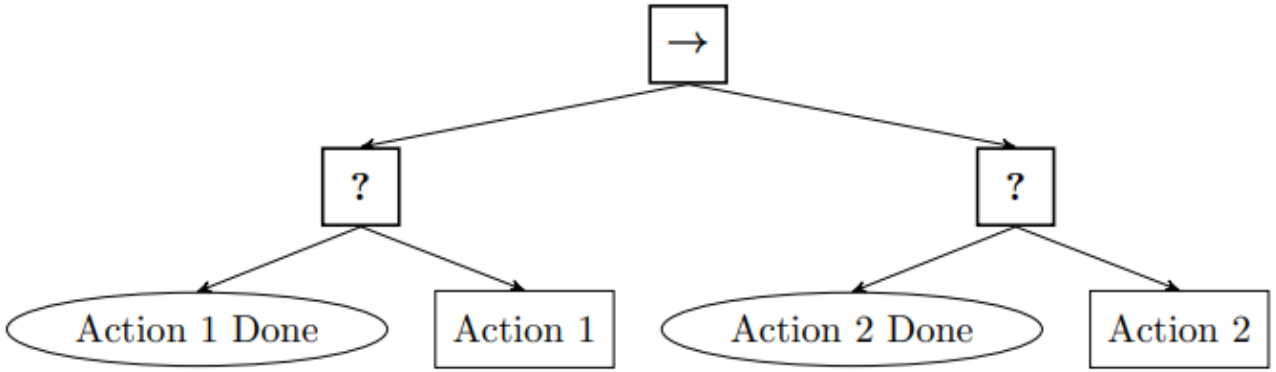
Elements in Behavior Tree

- Memory Node



Memory Node

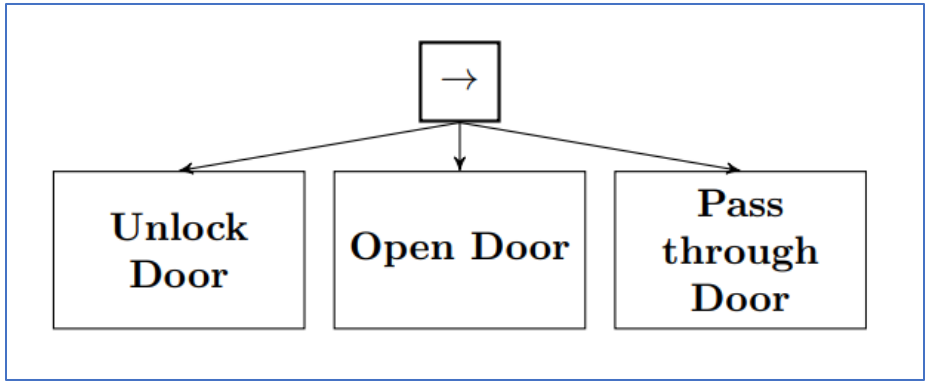
[The image is capture from the book[1].]



Normal Node

[The image is capture from the book[1].]

Start from simple and explicit

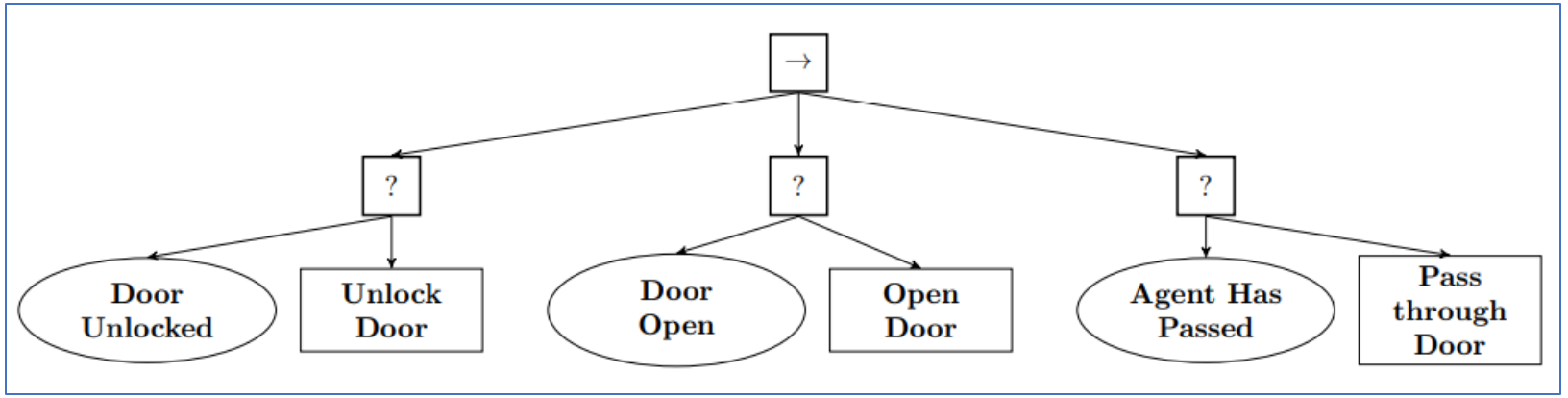


Simple sequence

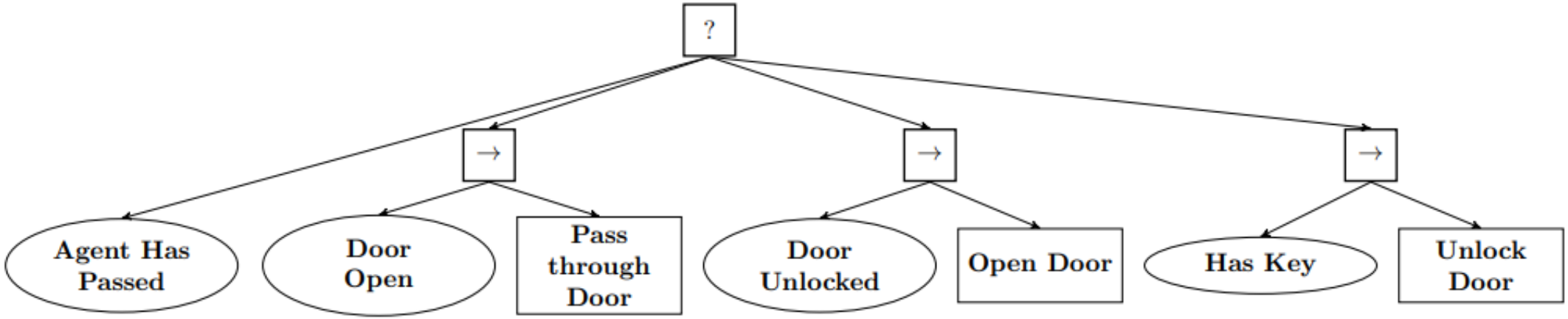
[The image is capture from the book[1].]

Add the explicitly condition

[The image is capture from the book[1].]



Giving more conditions and reorder

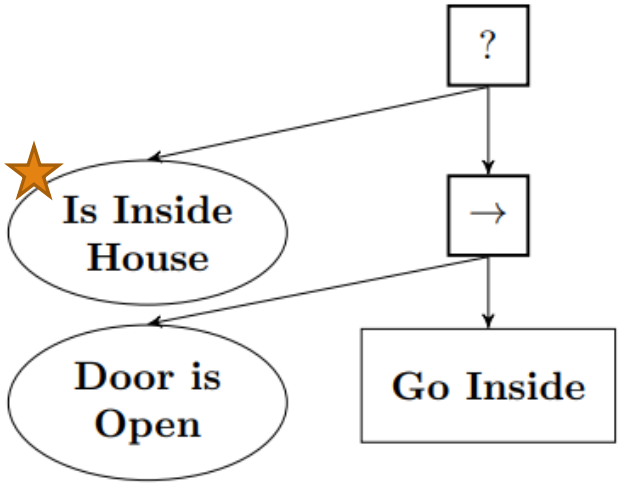


Reorder the state and add more condition

[The image is capture from the book[1].]

Creating Deliberative BTs

- Backchaining



Goal overview

[The image is capture from the book[1].]

Algorithm 4: Pseudocode of Backchaining Algorithm

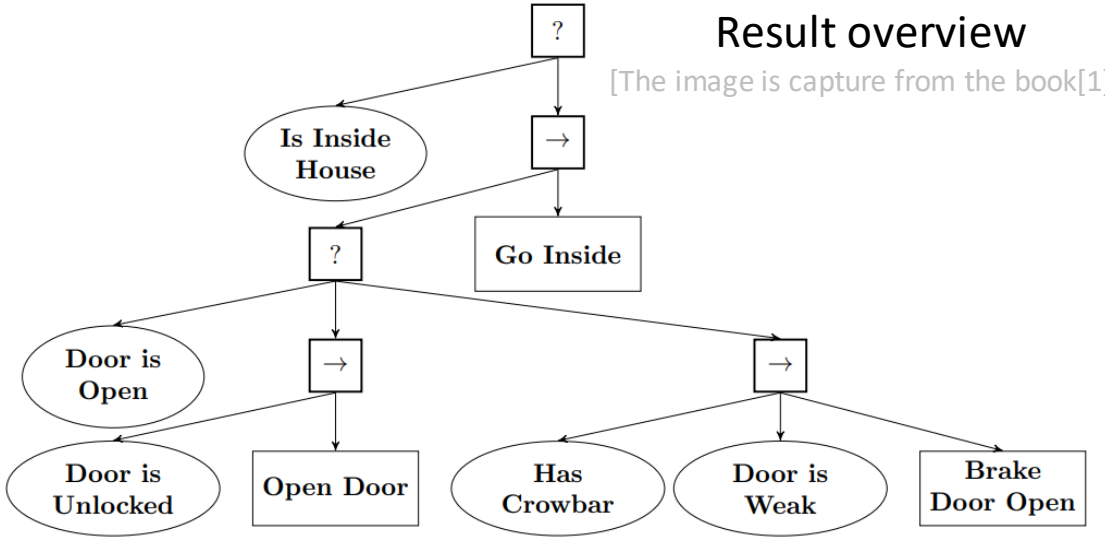
Data: Set of Goal Conditions C_i, and a set of PPAs
Result: A reactive BT working to achieve the C_is
1 Replace all C_i with PPAs having C_i as postcondition;
2 **while** *the BT returns Failure when ticked* **do**
3 replace one of the preconditions returning Failure (inside a PPA) with another complete PPA having the corresponding condition as postcondition, and therefore including at leaves one action to achieve the failing condition ;

Backchaining Algorithm

[The image is capture from the book[1].]

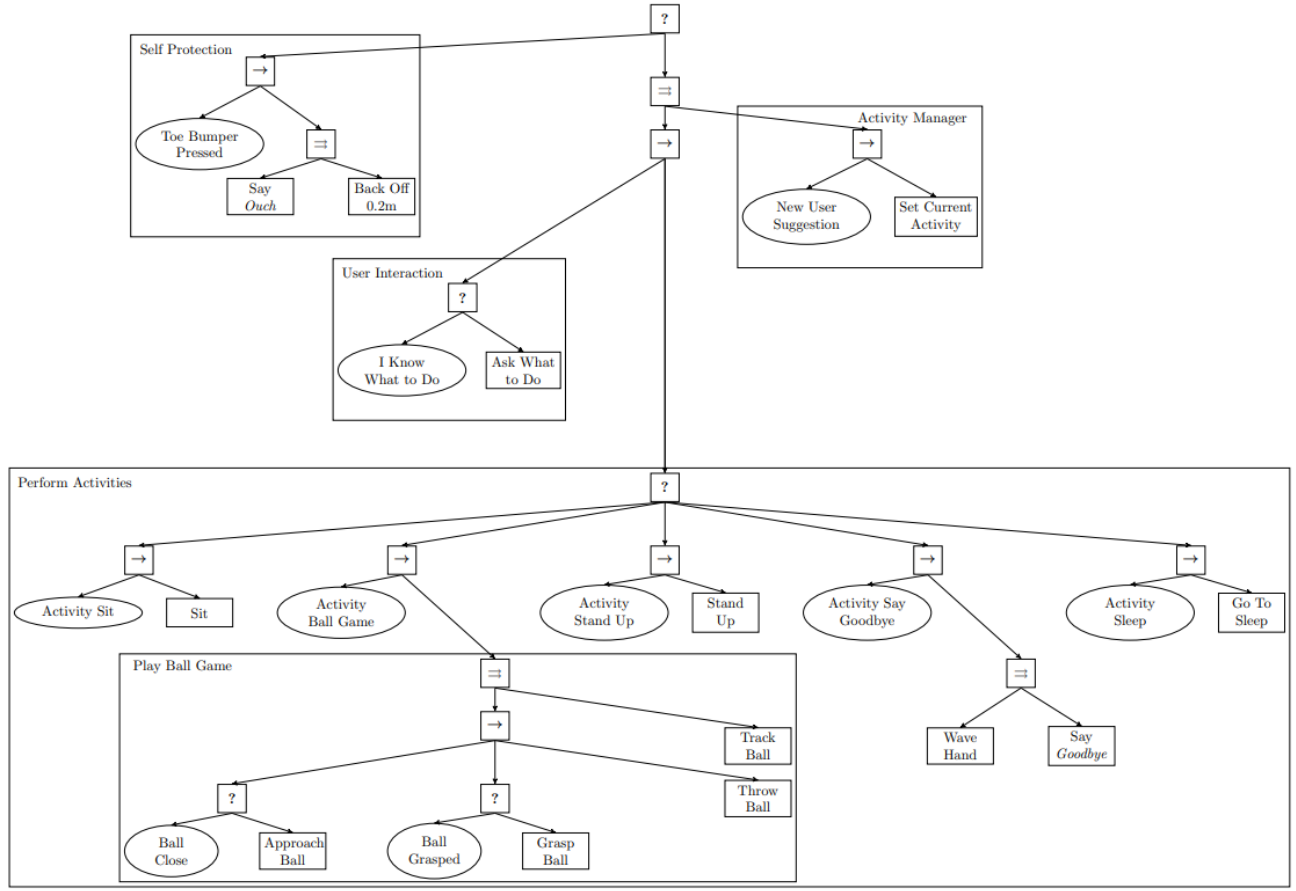
Result overview

[The image is capture from the book[1].]



Using proper granularity

- Overserve the reused part
- Encode each group of behavior with another leaf.

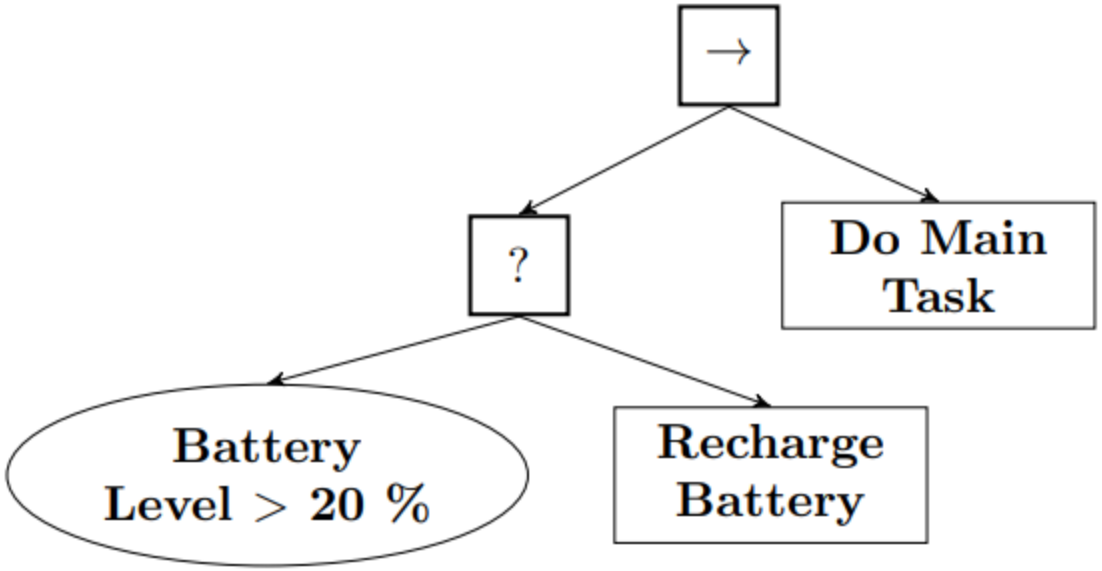


Robot activity manager

[The image is capture from the book[1].]

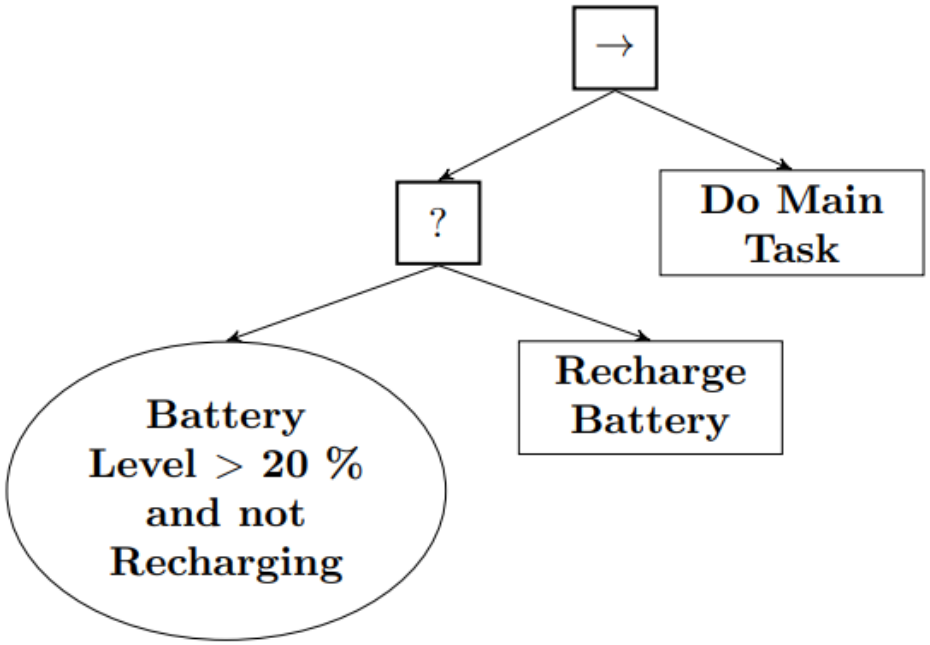
Thinking about safety

- Keep the battery level bigger than 20%



Safety Thinking

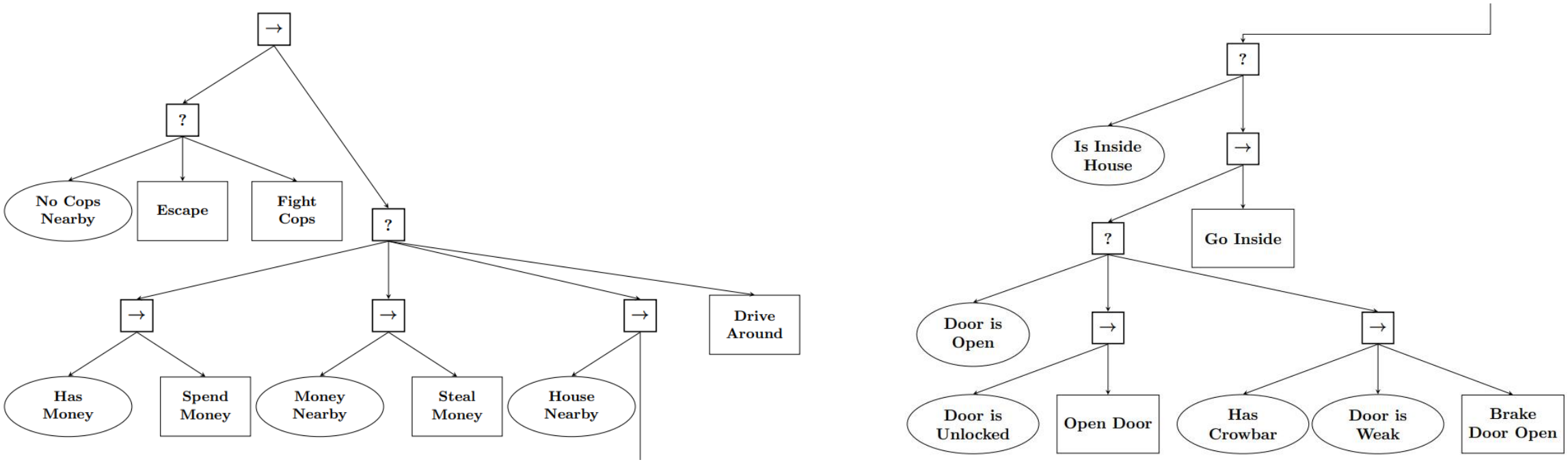
[The image is capture from the book[1].]



Prevent for recharging

[The image is capture from the book[1].]

Combination all aspect together



Final Result

[The image is capture from the book[1].]

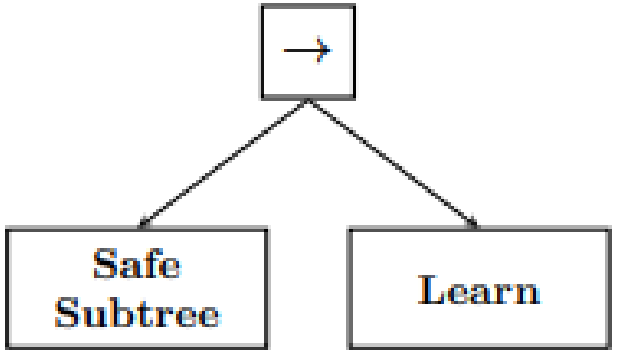
BT Create Demo

[3]

Behavior Trees and Machine Learning overview

Behavior Trees and Machine Learning

- Genetic Programming on BT (GP-BT)



Basic Concept

[The image is capture from the book[1].]

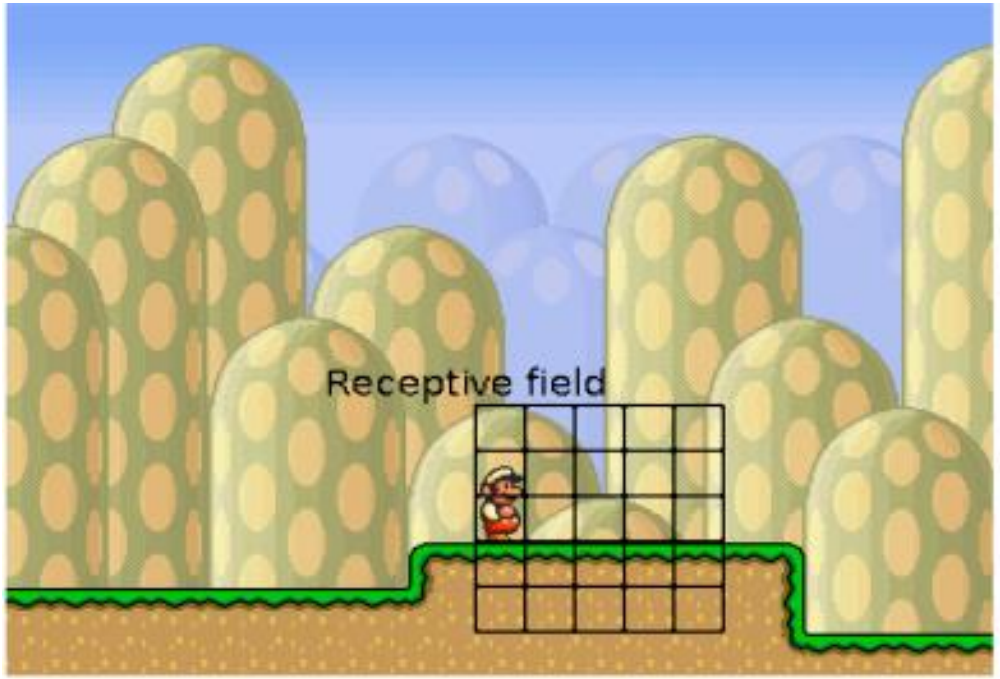
Algorithm 11: Pseudocode of the learning algorithm

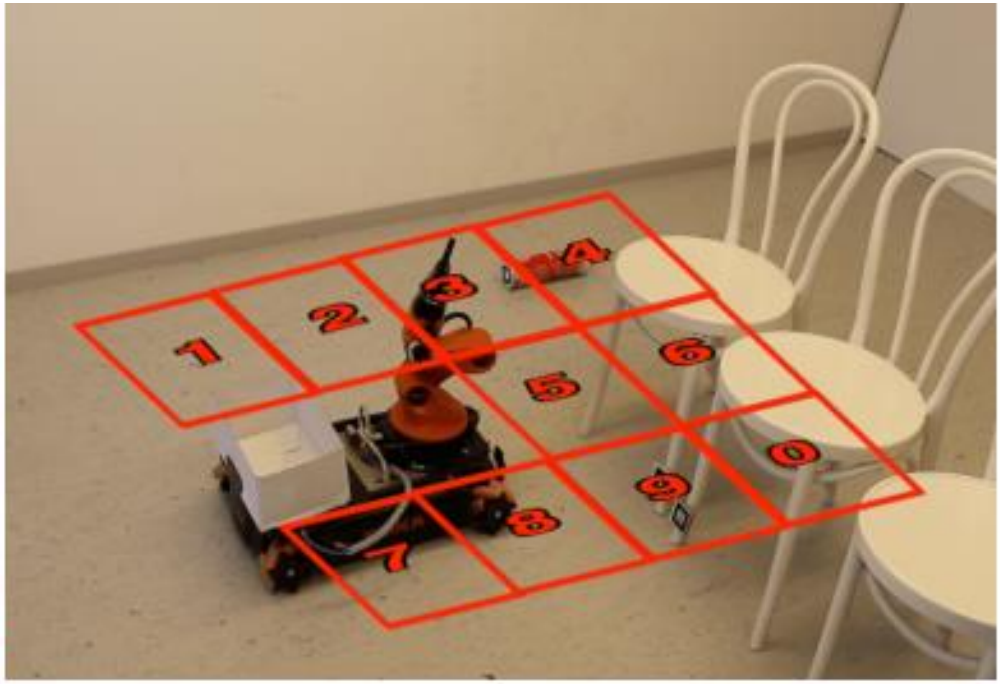
```
1  T ← "Action Learn"
2  do
3      Tick(SequenceNode(T_safe, T))
4      if IsExecuted(Action Learn) then
5          T_cond ← GetSituation()  %Eq(8.3)
6          T_acts ← LearnSingleAction(T)  %Eq(8.4)
7          if T_acts.NumOfChildren = 0 then
8              T_acts ← GetActionsUsingGP(T)  %Eq(8.5)
9          if IsAlreadyPresent(T_acts) then
10             T_cond_exist ← GetConditions(T_acts)
11             T_cond_exist ← Simplify(FallbackNode((T_cond_exist, T_cond))
12         else
13             T ← FallbackNode(SequenceNode(T_cond, T_acts), T)  %Eq(8.6)
14     ρ ← GetReward(SequenceNode(T_safe, T))
15 while ρ < 1;
16 return T
```

Learning Algorithm

[The image is capture from the book[1].]

Example of GP-BT





Examples

[The image is capture from the book[1].]

Q & A

THAK YOU FOR YOUR ATTENTION

Reference

[1] Colledanchise, M., & Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. CRC Press.

[2] Pacman Demo https://btirai.github.io/pacman

[3] Behavior Tree Cpp https://www.behaviortree.dev/